FoxPro Programming

What is Data?	Data can be anything like a name of person rollno of student or name of a city.
What is Information?	When Data is in meaning full way or related with each other its called Information.
What is Database?	Database is an organized collection of related information.
What is DBMS?	DBMS stands for Database Management System. The system which collect data from user and manage that data properly, is called Database Management system.
What is RDBMS?	RDBMS stands for Relational Database Management System. RDBMS is advance feature of DBMS. RDBMS provide the facility to create a relation between two tables.
What is FILE ?:-	In FoxPro File is a collection of records and fields. Row represent record and column represent field.
What is RECORD ?:-	Data is stored in one horizontal line of database is called record. In other words Record is a collection values in fields of table.
What is FIELD ?:-	Field is a basic part of any database column is also called field but in a perfect way <u>Field is a collection of data of same type and nature.</u>
What is Col. Width?	The number of digit you specify with each fieldname is called column width.
What is FoxPro?	FoxPro is one of the leading DBMS(Database Management System) software for pc. This is enhanced version of the FoxBASE+ Software FoxPro is also called RDBMS software.

DBMS software:-	1.Dbase	2.Foxpro	3.Foxbase	
RDBMS software:-	1.Oracle	2.Foxpro	3.MySQL server	4.MS Access
Extension of FoxPro:-	1) '.txt.' 2) '.dbf' 3) '.prg' 4) '.scr' 5) '.frx'			

❖ **Rules to create a new file:-**

1. Filename must start with an alphabets
2. Filename has maximum of eight characters.
3. Filename has only alphabets (A to Z, a to z, "_" underscore , 0 to 9 digit)
4. In Filename there must be no white space.
5. Filename must be unique. we can not type the same name as we type earlier.

❖ **Rules to create a New Field:-**

1. Fieldname must start with an alphabets
2. Fieldname has maximum of ten character long
3. Fieldname has only alphabets (A tO Z, a to z, "_"underscore , 0 to 9 digit)
4. In Fieldname there must be no white space.
5. Fieldname must be unique.

Difference between DBMS and RDBMS:-

DBMS	RDBMS
1. DBMS stands for Database Management System.	1. RDBMS stands for Relational Database Management System.
2. In DBMS we cannot create relation between two tables.	2. RDBMS provide facility to create relation between two tables.
3. DBMS may satisfy less than 7 to 8 rules of Dr.E.F.Codd.	3. DBMS may satisfy more than 7 to 8 rules of Dr.E.F.Codd.
4. In DBMS there is no security of data.	4. In RDBMS there are multilevel of security. (1) Logging in at o/s Level (2) Command Level (3) Object Level
5. Each table is given extension of DBMS	5. Many tables are grouped in one database in RDBMS.
6. DBMS use a 3GL.	6. DBMS use a 4GL.
7. Ex. of DBMS are Dbase, FoxBASE, Foxpro	7. Ex. of RDBMS is Oracle, Access, SQL server.

Data types available in FoxPro

1. Numeric:-

Numeric datatype is use to store numeric data into a field. We can store 0 to 9 digit, decimal point and plus or minus sign. A numeric field can hold upto 20 digit wide. A numeric fields can also have a decimal part. The decimal part can be upto 0 to 18 digit. To store a field like Roll_no, mobile_no, phone_no, salary, pincode etc numeric field type is used.

2. Float :-

Float datatype is similar to numeric. Difference between two is that for arithmetic calculation numeric datatype is used while float is used for scientific calculation. It can also hold upto 20 digit wide. We can store 0 to 9 digit, decimal point and plus or minus sign. To store the fields like Rate, percentage, average float is commonly used.

3. Character :-

Character datatype is use to store string type information. It can store A to Z , a to z alphabets 0 to 9 digit and underscore with special symbols etc. To store the fields like name, address, result etc character is used. Character is default datatype for all the fields. In Character data type we can store upto 254 character. Default size of character is 10 fix.

4. Date :-

Date datatype is use to store date in any field. The default format of date is (mm/dd/yy). The fix length of date is 8.To store the fields like Dob, doj, ex_date etc date field is used.

5. Logical :-

The length of logical field is 1. we can store ‘T’ or ‘F’ or ‘Y’ or ‘N’ in logical field. ’T’ or ‘Y’ for true and ‘F’ and ‘N’ for False. To store the information like pass/fail, absent/present etc. logical field type is used.

6. Memo :-

Memo field is special field of FoxPro in which you can store any amount of data. Memo field is use to store long textual information. The name of memo file has ‘.fpt’ extension. FoxPro allocates a ten bytes space in main dbf to store the location of memo data is in auxiliary memory.

7. General :-

General field is use only for FoxPro for window version. In general field we can store textual data, picture, sound etc.

Note :- In FoxPro picture field type is not used.

Simple commands of FoxPro:-

1. Creat command:-

Create command is use to create a new database file in FoxPro. We have to specify the name of file after the create keyword.

Syntax : creat <tablename>
Example : creat student

2. Clear command:-

Clear command is use to clear the screen.

Example : clear

3. Dir command:-

Dir command is use to see all the database file list of your current folder.

Example : Dir

4. Append command:-

Append command is use to append a new record in your table.

Syntax : append [blank]
Example : append

5. Display structure command:-

Display structure command is use to display a structure of your currently open table.

Example : disp stru

6. Modify structure command:-

Modify structure command is use to modify the structure of your currently open table.

Example : modi stru

7. Use command :-

Use command is use to open a your table in current work area. Use command is also use to close currently open table is well.

Syntax : use <tablename>[in <expn>/<workareaname>]
Example : creat student

8. Insert command:-

Insert command is use to insert a new record in database file. We can insert blank record through insert command. Before clause is use to insert a record before the current position.

Syntax : insert [Before][Blank]
Example : insert before

9. Display command:-

Displays the contents of records in the current table/.DBF and the results of expressions.

```
Syntax      :  Display[[FIELDS] <field list>]
                      [<scope>]
                      [FOR <expL1>]
                      [WHILE <expL2>]
                      [OFF]
```

[[FIELDS] <field list>] Clause:- All fields from the table/.DBF are displayed unless you include FIELDS <field list>. Then, only the fields included in the field list are displayed.

[<scope>] clause:- The scope clauses are:

[1] ALL [2] NEXT <expN> [3]RECORD <expN> [4] REST

You can specify any scope of records to display. Only the record that fall with in the range of records specified by the scope is displayed. The default scope for DISPLAY is the current record next one.

[FOR<expL1>] clause :- Including the FOR clause lets you conditionally display records, filtering out undesired records. If the FOR clause is included, only the records that satisfy the logical condition <expL1> are displayed.

[WHILE <expL2>] clause:-

If the WHILE clause is included, records are displayed for as long as the logical expression <expL2> evaluates to true.

[OFF] clause:-

Include OFF to suppress record number display. If OFF is omitted, the record number is displayed before each record.

10. List command :-

List command displays all the records of currently open table and the result of expression.

```
Syntax:-        List  [[FIELDS] <field list>]
                      [<scope>]
                      [FOR <expL1>]
                      [WHILE <expL2>]
                      [OFF]
```

[[FIELDS] <field list>] Clause:-

All fields from the table/.DBF are displayed unless you include FIELDS <field list>. Then, only the fields included in the field list are displayed.

[<Scope>] clause:-

The scope clauses are:

[1] ALL [2] NEXT <expN> [3]RECORD <expN> [4] REST

You can specify any scope of records to display. Only the record that fall with in the range of records specified by the scope are displayed. The default scope for LIST is the ALL record.

FOR<expL1>] clause:-

Including the FOR clause lets you conditionally display records, filtering out undesired records. If the FOR clause is included, only the records that satisfy the logical condition <expL1> are displayed.

[WHILE <expL2>] clause:-

If the WHILE clause is included, records are displayed for as long as the logical expression <expL2> evaluates to true.

[OFF] clause:-

Include OFF to suppress record number display. If OFF is omitted, the record number is displayed before each record.

11. Edit /Change command:-

Edit or Change command is use to edit the record which are inserted before by us.

```
Syntax:-        Edit  [[FIELDS] <field list>]
                      [FOR <expL1>]
                      [WHILE <expL2>]
                      [FREEZE<fieldname>]
                      [NO APPEND][NO DELETE]
                      [NO CHANGE | NO MODIFY]
```

[[FIELDS] <field list>] Clause:-

If FIELDS is included in CHANGE/EDIT the listed fields are displayed in the order they appear in <field list>. If FIELDS isn't included, all fields in the table/.DBF are displayed in the order they appear in the table/.DBF structure..

[FOR<expL1>] clause :-

Including the FOR clause lets you conditionally display records, filtering out undesired records. If the FOR clause is included, only the records that satisfy the logical condition <expL1> are displayed in the Change window.

[WHILE <expL2>] clause :-

If the WHILE clause is included, records are displayed in the Change window. For as long as the logical expression <expL2> evaluates to true.

[FREEZE <field name>] Clause :-

Include the FREEZE clause to allow changes to be made to a single field in the Change window. You specify this field with <field name>. The remaining fields are displayed and cannot be edited.

[NOAPPEND] Clause :-

If you include NOAPPEND in CHANGE, you can't add records to the table/.DBF by pressing Ctrl+N or choosing Append Record from the Browse menu.

[NODELETE] Clause :-

Including NODELETE prevents records from being marked for deletion from within a Change window. By default, a record can be marked for deletion by pressing Ctrl+T, choosing Toggle Delete from the Browse menu or clicking in the leftmost column of the record to be deleted.

[NOEDIT or NOMODIFY] Clause :-

Including NOEDIT or NOMODIFY prevents you from modifying the table/.DBF. NOEDIT and NOMODIFY are identical. If you include either clause, you can browse or search the table/.DBF, but you cannot be edit it. However, you can append and delete records. Include NOAPPEND and NODELETE to prohibit the addition and deletion of records.

<u>12.Browse command:-</u>

Browse command opens a browse window to change records of currently open data base file like a tabular format.

```
Syntax:-  Browse  [FIELDS] <field list>]
                  [FOR <expL1>]
                  [WHILE <expL2>]
                  [FREEZE<fieldname>]
                  [WIDTH <expN>]
                  [FONT <expC>[,<expN>]]
                  [STYLE<expC>]
                  [NO APPEND][NO DELETE]
                  [NO CHANGE | NO MODIFY]
```

[[FIELDS] <field list>] Clause:-

If FIELDS is included, the listed fields are displayed in the order specified in <field list>. If FIELDS isn't included, all fields in the table/.DBF are displayed in the order they appear in the table/.DBF structure.

[FOR<expL1>] clause :-

Including the FOR clause lets you conditionally display records in a Browse window, filtering out undesired records. If you include the FOR clause, only records for which the logical expression <expL1> is true are displayed in the Browse window.

[WHILE <expL2>] clause :-

If the WHILE clause is included, records are displayed in the Change window.For as long as the logical expression <expL2> evaluates to true.

[FREEZE <field name>] Clause :-

Include the FREEZE clause to permit changes to be made to only one field in the Browse window. You specify this field with <field name>. The remaining fields are displayed and cannot be edited.

[FONT <expC>[,<expN>]]Clause:-

The character expression <expC1> is the name of the font, and the numeric expression <expN1> is the font size. For example, the following clause specifies 16-point Roman font for the fields displayed in a Browse window:

<u>Example:-</u> FONT 'ARIAL',16

If you include the FONT clause but omit the font size <expN1>, a 10 point font is used in the Browse window. If you omit the FONT clause, 8 point MS Sans Serif is used. if the font you specify is not available, Windows substitutes a font with similar font characteristics.

[STYLE<expC>]clause:-

In FoxPro for Windows, include the STYLE clause to specify a font style for fields displayed in the Browse window. The styles that are available for a font are determined by Windows. If the font style you specify is not available, Windows substitutes a font style with similar characteristics. The font style is specified with <expC2>.If the STYLE clause is omitted, the standard font style is used. Character which is use are follow with the style name.

B	Bold,	I	Italic
N	Normal,	O	Outline
S	Shadow,	-	Strikeout
U	Underline		

You can include more than one character to specify a combination of font styles. The following example open a Browse window and uses a 15 point underlined System font:

The following Clauses are works same as in edit command. so see them.([NOAPPEND] [NODELETE] [NOEDIT or NOMODIFY])

13. Delete command :-

Delete command is use to make a mark of Delete of record in currently open data base file. Records marked for deletion are not physically removed from the table until PACK is issued. Records marked for deletion can be recalled (unmarked) with RECALL command. Records can also be marked for deletion from within a Browse, Change or Edit window or from the Record menu.

```
Syntex:-    DELETE [<scope>]
                   [FOR <expL1>]
                   [WHILE <expL2>]
```

[<scope>] clause:-

The scope clauses are: [1] ALL [2] NEXT <expN> [3]RECORD <expN> [4] REST

You can specify a scope of records to mark for deletion. Only the records that fall within the range of records specified by the scope are marked for delet.The default scope for DELETE is the current record means next one.

[FOR<expL1>] clause :-

Including the FOR clause lets you conditionally delete records. If the FOR clause is included, only the records that satisfy the logical condition <expL1> are marked for deletion.

[WHILE <expL2>] cluase :-

If the WHILE clause is included, records are marked for deletion for as long as <expL2> evaluates to true (.T.).

14. Recall command:-

Recall command is use to unmark the record which are marked for deletion by Delete command. Before using recall command there must be zap or pack commands are not used.

```
Syntax:-    Recall [<scope>]
                   [FOR <expL1>]
                   [WHILE <expL2>]
```

[<scope>] clause:-

The scope clauses are: [1] ALL [2] NEXT <expN> [3]RECORD <expN> [4] REST

You can specify a scope of records to recall. Only the records that fall within the range of records specified by the scope are recalled. The default scope for DELETE is the current record means next1

[FOR<expL1>] clause :-

If the FOR clause is included, only the records for which <expL1> evaluates to true (.T.) are recalled. Including a FOR clause lets you conditionally recall records, filtering out undesired records.

[WHILE <expL2>] clause :-

If the WHILE clause is included, records are recalled for as long as <expL2> evaluates to true.

15. Pack command:-

PACK command is use to permanently removes all records marked for deletion in the current table/.DBF.

Synaex:- pack

16. Zap command:-

ZAP command is use to permanently removes all records from current table/.DBF. by leaving just structure of a table.

Example:- zap

17. Skip command :-

Skip command is use to Moves the record pointer forward or backward in the current or specified table/.DBF.

```
Syntax:-     Skip <expN>[in <expN> or <expC>]
Example:-    skip 2
```

18. Go/Goto command:-

Go/Goto command is use to Positions the record pointer on the specified record number in a table/.DBF.

```
Syntax:-     Go <expN>[in <expN> or <expC>]
             GO TOP | BOTTOM
Example:-    Go top
```

19. Replace command :-

Replace command is use to insert a value in field. Replace is also use to change the value of fields in table/.DBF.

```
Syntex:-     REPLACE <field1> WITH <expr1>
                          [ADDITIVE]
                          [, <field2> WITH <expr2>
                          [ADDITIVE]] ...
                          [<scope>]
                          [FOR <expL1>]
                          [WHILE <expL2>]
```

RPLACE <field1> WITH <expr1>[,<field2> WITH <expr2>]:

Data in a field1 is replaced with the value of exp1,the data in field2 is replaced with exp2.

ADDITIVE clause:-

The ADDITIVE clause applies to replacements in memo fields only. If ADDITIVE is included, replacements to memo fields are appended to the end of the memo fields. If ADDITIVE is not included, the memo field is overwritten with the value of the expression.

[<scope>] clause:-

The scope clauses are: [1] ALL [2] NEXT <expN> [3]RECORD <expN> [4] REST

You can specify a scope of records to replace. Only the records that fall within the range of records specified by the scope are replaced. The default scope for REPLACE is the current record NEXT 1.

[FOR<expL1>] clause :-

if the FOR clause is included, the specified fields are replaced only in records for which <expL1> evaluates to true. Including FOR lets you conditionally replace records, filtering out undesired records.

[WHILE <expL2>] clause :-

If the WHILE clause is included, fields in records are replaced for as long as the logical expression <expL2> evaluates to true.

20. Locate command :-

Locate command is use to search the record sequentially in table/.DBF. for the first record that matches a given logical exp.

```
Syntex:-     LOCATE     FOR <expL1>
                        [<scope>]
                        [WHILE <expL2>]
```

Locate command return the record number where it find the record with the true condition. Default scope of locate command is all records. Here for clause, scope clause and while clause works same as in other command. Plz to read from other command.

21. Sort command :-

Sort command is useful to sort records of the current database file and stored in a new database file Each sort command generate a new database file.

```
Syntex:-     SORT TO <file> ON <field1>
                            [/A | /D] [/C]
                            [, <field2> [/A | /D] [/C] ...]
                            [ASCENDING | DESCENDING]
                            [<scope>] [FOR<expL1>]
                            [WHILE <expL2>]
                            [FIELDS <field list>]
```

<file> Clause:-

When a table is sorted, a new table is created. <file> specifies the name of the new table file. FoxPro assumes a .DBF file name extension for tables. A '.DBF' extension is automatically assigned if the file name you include doesn't have an extension.

ON <field1> Clause:-

You must include a name of a field (<field1>) from the current table. The contents and data type of the field determine the order of the records in new table. By default, the sort is done in ascending order. You can't sort on memo or general fields. You can include additional field names (<field2>, <field3>) to further order the new table. The first field <field1> is the primary sort field, the second field <field2> is the secondary sort field, and so on.

[/A | /D] [/C] Clause:-

For each field you include in the sort you can specify an ascending or descending sort order. The /A option specifies an ascending order for the field, and the /D option specifies a descending order. /A or /D can be included with any type of field. By default, the field sort order for character fields is case sensitive If you include the /C option after the name of a character field, case is ignored. You can combine the /C option with the /A or /D option. Use one slash (/AC or /DC) if you include the /C option with /A or /D.

ASCENDING | DESCENDING Clause:-

You can specify a sort order for all sort fields not followed by the /A or /D options. All sort fields are sorted in ascending order if you include ASCENDING. All sort fields are sorted in descending order if DESCENDING is included. The sort fields default to an ascending order if ASCENDING or DESCENDING isn't included.

[<scope>] clause:-

The scope clauses are: [1] ALL [2] NEXT <expN> [3]RECORD <expN> [4] REST

You can specify a scope of records to sort. Only the records that fall within the range of records specified by the scope are sorted. The default scope for SORT is ALL records.

[FOR<expL1>] clause :-

If the FOR clause is included, only the records for which <expL1> evaluates to true (.T.) are Sorted. Including a FOR clause lets you conditionally Sorting the records, filtering out undesired records.

[WHILE <expL2>] clause :-

If the WHILE clause is included, records are sorted for as long as <expL2> evaluates to true.

FIELDS <fields list>clause:-

The new table that SORT creates can contain a subset of fields from the original table. All fields from the original table are included in the new table if the FIELDS clause isn't included. Include FIELDS and a <fields list> to include specific fields from the original table.

22. Index command :-

Indexing a database create an index file which is almost similar to the index at the end of the book. As you use the index of a book to locate or to search the page number where a particular record or topic is discussed, FoxPro also use the index file for the same purpose. Unlike a sorting Indexing a database file does not duplicate the file. It just stores the index fields and corresponding record number in the index file. So it is smaller than sorted file.

Syntax:-

```
INDEX ON <expr> TO <idx file> | TAG <tag name> [OF <cdx file>]
            [FOR <expL>]
            [COMPACT]
            [ASCENDING | DESCENDING]
            [UNIQUE]
            [ADDITIVE]
```

<expr>

The index expression <expr> can include the name of a field or fields from the current table/.DBF. An index key based on the index expression is created in the index file for each record in the table/.DBF.

TO <idx file> <idx file> specifies the name of index file.

TAG <tag name> [OF <cdx file>] clause:-

Tag<tagname> clause is use to create a structural compound index file.

[OF <cdx file>] clause:- Of <cdx file> specify the name compound index file.

UNIQUE clause:-

UNIQUE clause is use to creat a unique index file. Unique clause prevent to add duplicate record in index file at time of creating index file. By default all the records from the table are included for indexing. But if you include unique option the duplicate record will not be display.

ADDITIVE clause:-

When you create an index file for a table with index command any previous open index file is close. if the Additive clause is included previously open file remains open.

COMPACT clause:-

you include compact option with the index file that you create will be in the compact format. It may have less size than the original index file.

[<scope>] clause:-

The scope clauses are: [1] ALL [2] NEXT <expN> [3]RECORD <expN> [4] REST

You can specify a scope of records to index. Only the records that fall within the range of records specified by the scope are indexed. The default scope for INDEX is ALL records.

[FOR<expL1>] clause :-

If the FOR clause is included, only the records for which <expL1> evaluates to true (.T.) are Sorted. Including a FOR clause lets you conditionally Sorting the records, filtering out undesired records.

[WHILE <expL2>] clause :-

If the WHILE clause is included, records are indexing for as long as <expL2> evaluates to true.

23. Erase command :-

Erase command is use to delete any database file. You have to specify the name of file that you want to delete.

Syntax:- Erase <tablename>
Example:- Erase student

24. Copy command :-

Copy command is use to copy one folder to another folder.

Syntax:- copy <source path+tablename> to <destination path>
Example:- copy e:\pgdca\student.dbf to e:\bca\student.dbf

25. Rename command :-

Rename command is use to change the name of your file.

Syntax:- rename <old tablename> to <new tablename>
Example:- rename student to pgstudent

26. Seek command :-

Seek command is use to search the record in index file. Seek search only in indexed file with an expression.

Syntax : seek<expR>
Example : seek "rus"

You can use SEEK only with indexed tables/.DBFs, and you can search only on the index key expression. The match must be exact unless SET EXACT is OFF. If SEEK finds a matching record, RECNO( ) returns the record number of the matching record, FOUND( ) returns true and EOF( ) returns false. If no match is found, RECNO( ) returns the total number of records in the table/.DBF plus 1, FOUND( ) returns false and EOF( ) returns true.

27. Close all command :-

Close all command is use to close all open programs, databasefile, indexfile etc.

Syntax:- Close all
Example:- Close all

28. Scatter command :-

Scatter command is use to Copy data from the current record to an array or a set of memory variables.

Syntax:- Scatter [Fields<fieldslist>] to <arrayvariable>

While scatter command is use it is necessary that array variable have the same size as many fields in the database file.If the array variable is not created before scatter command automatically create an array variable as the size as many records in the table/Database file.

Example:-
```
use student
Go 1
Scatter to rec1
Scatter name,city,dob to rec2
```

29. Gather command :-

Gather command is use Store data from a memory variable array or a set of memory variables to the current record of the active table/.DBF.While Gather command is used its necessary database file is open.

Syntax:- Gather from <arrayvariable>[FIELDS<fieldslist>]

If data is being copied from an array, the elements of the array are copied, starting with the first element,into the corresponding fields of the record. The contents of the first array element is copied to the first field of the record; the contents of the second array element is copied to the second field and so on.If the array has fewer elements than the table/.DBF has fields, the additional fields are ignored. If the array has more elements than the table/.DBF has fields, the additional array elements are ignored.

Example:-
```
use student
Go bottom
Gather from  test1
Gather from test1 fields name,city,dob
```

Functions of FoxPro

1. Numeric Function **2. String Function**
3. Date & Time Function **4. Array Function**

1. Numeric Function:-

(1) Abs():- This function returns the absolute value of the specified numeric expression.

Syntax	:	Abs(<expN>)
Function Returns	:	Numeric
Example	:	Abs(-25)
Output	:	25

(2) Int():- This function returns the Ineger value from the specified numeric expression.

Syntax	:-	Int(<expN>)
Function Returns	:-	Numeric
Example	:-	Int(25.25)
Output	:-	25

(3) Floor():- This function returns the nearest integer value that is less than or equal to the specified numeric expression.

Syntax	:	Floor(<expN>)
Function Returns	:	Numeric
Example	:	Floor(51.85)
Output	:	51

(4) Ceil():- This function returns the nearest integer value that is greater than or equal to the specified expression.

Syntax:-	Ceil(<expN>)
Function Returns:-	Numeric
Example:-	Ceil(27.35)
Output:-	28

(5) Mod():- This function returns the remainder value by moduling expression1 to expression2.

Syntax	:-	Mod(<expN1>,<expN2>)
Function Returns	:-	Numeric
Example	:-	Mod(5,2)
Output	:-	1

(6) Round():- This function Return a numeric expression rounded to a specified number of decimal places.

Syntax:-	Round(<expN1>,<expN2>)	
Function Returns:-	Numeric	
Example:-	Round(25.49)	Round(25.51)
Output:-	25	26

(7) Min():- This function Returns the expression with the lowest ASCII or numeric value or the earliest date in a list of character, numeric or date expressions.

Syntax:-	Min(<expN1>,<expN2>[<expN>……])
Function Returns:-	Numeric,date,Character
Example:-	Min(10,20,5,85,50)
Output:-	5

(8) Max():- Returns the expression with the highest ASCII or numeric value or the latest date from a list of character, numeric or date expressions.

Syntax:-	Max(<expN1>,<expN2>[<expN>……])
Function Returns:-	Numeric,date,Character
Example:-	Min(10,20,5,85,50)
Output:-	85

(9) Sqrt():- This function Return the square root of specified expression.

Syntax:-	Sqrt(<expN1>)
Function Returns:-	Numeric
Example:-	Sqrt(25)
Output:-	5

(10) Log():- This function Return the simple natural loqarism value of specified numeric expression.

Syntax:-	Log(<expN>)
Function Returns:-	Numeric
Example:-	Log(10)
Output:-	2.30

(11) Log10():- This function Return the simple natural log base10 value of specified numeric exp.

Syntax:-	Log10(<expN>)
Function Returns:-	Numeric
Example:-	Log10(10)
Output:-	1.00

2.Array Function:-

(1) Asort():- This function is use to Sorts elements in an array in ascending or descending order.

Syntax:-	ASORT(<array>[, <expN1>[, <expN2>[, <expN3>]]])
Function Returns:-	Numeric

All elements included in the sort must be of the same data type (character, numeric, date or logical). One-dimensional arrays are sorted by their elements; two-dimensional arrays are sorted by their rows. When a two-dimensional array is sorted, the order of the rows in the array is changed so the elements in a column of the array are in ascending or descending order.If the sort is successful, 1 is returned; otherwise -1 is returned.<array> specify the name of array memory Vriable. <expn1> specifies the starting element number from where to start sorting,<expn2>specifies the number of array how many records are to be sorting,<expn3>specifies sorting order(0 for ascending order and 1 for descending).

```
Example    :  declare m1[4]={'c','b','d','a'}
              ?asort(m1,1,4,1)
Output     :  1
```

(2) Alen():- This function Returns the number of elements, rows or columns in an array.

Syntax:-	ALEN(<array>[, <expN>])
Function Returns:-	Numeric

<array> If you include only the array name <array>, ALEN( ) returns the number of elements in the array. <expN> can be 0, 1 or 2. If <expN> is 0 function returns the total no.of element if the <expN> is 1 function returns the total no.of rows in an array and if the <expN> is 2 returns the total no.of column in an array.<expN>is use when array is two dimensional.

```
Example:-   declare m1[4]={'c','b','d','a'}
            ?Alen(m1)
Output:-    4
Example:-   declare m2[2,5]                declare m2[2,5]
            ?Alen(m1,1)                    ?Alen(m1,2)
Output:-    2                              5
```

(3) Aelement(): This function Returns the number of an array element from the element's subscripts. This function is commenly used in two dimensional array.

Syntax:-	AELEMENT(<array>, <expN1>[, <expN2>])
Function Returns:-	Numeric

<array>Include the name of the array whose element number you want to return. <expN1> Specify the row subscript with <expN1>. If the array is one-dimensional, AELEMENT() identically returns <expN1>. <expN2> Specify the column subscript with <expN2>. If the array is two-dimensional, include both <expN1> and <expN2>. If you include just <expN1>, the element number is returned

until <expN1> exceeds the number of rows in the array, then the error message "Subscript out of bounds" is displayed.

```
Example:-   declare m1[5]            declare m2[2,5]
            ?Aelement(m1,3)          ?Aelement(m1,2,2)
Output:-    3                        7
```

(4) Ains():- This function is use to Inserts an element into a one-dimensional array, or a row or column into a two-dimensional array. AINS( ) returns 1 if the element, row or column is successfully inserted.

Syntax:- AINS(<array>, <expN>[, 2])

Function Returns:- Numeric

To insert an element into a one-dimensional array, include the array name <array> and the number of the element <expN> where the insertion occurs. The new element is inserted just before element <expN>. To insert a row into a two-dimensional array, include the array name <array> and the number of the row <expN> where the insertion occurs. The new row is inserted just before row <expN>. To insert a column into a two-dimensional array, include the array name, the column number <expN> where the insertion occurs and the argument 2.The new column is inserted just before the column specified with<expN>.

```
Example:-   declare m1[5]={‘a’,’c’,’b’,’g’,’h’}
            ?Ains(m1,3)
            Disp memo like m1
Output:-    m1[1]=’a’   m1[2]=’c’   m1[3]=’.f.    m1[4]=’b’   m1[5]=’g’
```

Thus when we insert a new element in an array computer automatically drop the lat element and creat a blank space at specified no and default value is as .f. is assigned.

(5) Adel():- This function is use to Deletes an element from a one-dimensional array or a row or column from a two-dimensional array. If the element, row or column is successfully deleted, 1 is returned.

Syntax:- ADEL (<array>, <expN>[, 2])

Function Returns:- Numeric

Deleting an element, row or column from an array doesn't change the size of the array; instead, the trailing elements, rows or columns are moved to the start of the array, and the last element, row or column in the array is set to a logical false (.F.).

```
Example:-   declare m1[5]={‘a’,’c’,’b’,’g’,’h’}
            ?Adel(m1,3)
            Disp memo like m1
Output:-    m1[1]=’a’   m1[2]=’c’   m1[3]=’g’    m1[4]=’h’   m1[5]=’.f.’
```

(5) Asubscript():- This function Returns the row or column subscript of an element from the element's number. array.

Syntax:- ASUBSCRIPT(<array>, <expN1>, <expN2>)

Function Returns:- Numeric

If the array is one-dimensional, include the element number in <expN1> and 1 in <expN2>. ASUBSCRIPT( ) identically returns <expN1>.If the array is two-dimensional you must include both the element number <expN1> and a value of 1 or 2 in <expN2>. Specifying 1 in <expN2> returns the row subscript of the element, and specifying 2 returns the column subscript.

```
Example:-   declare m1[5]={‘a’,’c’,’b’,’g’,’h’}
            ?Asubscript(m1,3,1)      ?Asubscript(m1,5,1)
Output:-    3                        5
Example:-   declare m1[3,2]={‘a’,’c’,’b’,’g’,’h’,’j’}
            ?Asubscript(m1,3,1)      ?Asubscript(m1,5,)
Output:-    2                        3
```

(5) Acopy(): This function is use to Copies elements from one array to another array.This function returns the number of elements copied to the destination array.

Syntax:-	ACOPY(<array1>, <array2>[, <expN1>[, <expN2>[,<expN3>]]])
Function Returns:-	Numeric

3.Date and Time Function:-

(1) Date():- Date function returns the current date in mm/dd/yy format. This function is use to print date on report. Foxpro display the date as set above in computer.

Syntax	:	Date()
Example	:	? Date()
Output	:	01/21/08
Function returns	:	Date

(2) Year():- Year function returns the numeric year from the date expression.This function is always display the year with century. Foxpro display the as set above in computer.

Syntax:-	Year(<exprDate>)
Example:-	? Year(date())
Output:-	2008
Function returns:	Numeric

(3)Month():- Month function returns the numeric month from the date expression. Foxpro display the month as set above in computer.

Syntax:-	Month(<exprDate>)
Example:-	? Month(date())
Output:-	1
Function returns:	Numeric

(4)Cmonth():- Cmonth function returns the name of month in character from the date expression. Foxpro display the monthname as set above in computer.

Syntax:-	Cmonth(<exprDate>)
Example:-	? Cmonth(date())
Function returns:	Character
Output:-	January

(5)Day():- Day function returns the numeric day value from the date expression. Foxpro display the day as set above in computer.

Syntax:-	Day(<exprDate>)
Example:-	? Day(date())
Output:-	21
Function returns:	Numeric

(6)CDOW():- CDOW means character day of week.This function returns the character day from the date expression. Foxpro display the day as set above in computer.

Syntax:-	CDOW(<exprDate>)
Example:-	? CDOW(date())
Output:-	Monday
Function returns:	Character

(7)DOW():- DOW means Day of Week. This function returns the Numeric day value from the date expression. Foxpro display the day as set above in computer.(0 for Sunday and 6 for Saturday)

Syntax:-	DOW(<exprDate>)
Example:-	? DOW(date())
Output:-	1
Function returns:	Numeric

(8) Gomonth():- Gomonth function Returns the date that is a specified number of months before or after a given date.Foxpro display the day as set above in computer.

Syntax:- Gomonth(<exprDate>,<expN>)
Example:- ? Gomonth(date(),3)
Output:- 04/21/08
Function returns: Date

(9)DtoC():- DtoC means Date to character. This function is use to change date to char format.

Syntax:- DtoC(<exprDate>)
Example:- d1=date()
Disp memo like d1 (oputput:-d1 date ‘01/21/08’)
D1=DtoC(d1)
Disp memo like d1 (oputput:-d1 Character ‘01/21/08’)
Function returns:- Character

(10) CtoD():- CtoD means Character to Date. This function is use to change character type date in date format.

Syntax:- CtoD(<expC>)
Example:- d1=’05/11/07’
Disp memo like d1 (oputput:-d1 Charater ‘05/11/07’)
D1=CtoD(d1)
Disp memo like d1 (oputput:-d1 Date ‘05/11/07’)
Function returns:- Date

(11) Time():- Time function always return the current time as set in computer. This function Returns the current system time in 24-hour, eight-character string (HH:MM:SS) format. This function is use to print the time of printout in reports.

Syntax:- Time()
Example:- ?Time()
Output:- 08:52:25
Function returns:- Character

(12) Second():- Second function always return the seconds that were passed between 12pm to current time as set in computer.This function returns the value in floating point.

Syntax:- Second()
Example:- ?Second()
Output:- 32128.830
Function returns:- Numeric

(13)Sys(2):- Sys function always return the seconds that were passed between 12pm to current time as set in computer.This function returns the value in Integer format.

Syntax:- Second()
Example:- ?Second()
Output:- 32128
Function returns:- Numeric

4. String Function:-

(1) Chr():- This function Returns the character associated with the specified numeric ASCII code.

Syntax:- Chr(<exp N>)
Example:- ?Second(65)
Output:- A
Function returns:- Character

(2) Asc():- This function Returns the ASCII code for the leftmost character in a character express

Syntax:- Asc(<exp C>)
Example:- ?Second('a')
Output:- 97
Function returns:- Numeric

(3) Val():- This function Returns a numeric expression from a specified character expression composed of numbers.

Syntax:- Val(<exp C>)
Example:- d1="55"
Disp memo like d1 (oputput:-d1 Character '55')
D1=val(d1)
Disp memo like d1 (oputput:-d1 Numeric 55)
Function returns:- Numeric

(4) Str():- This function Returns the character string equivalent to a specified numeric express.

Syntax:- Str(<exp N>)
Example:- d1=55
Disp memo like d1 (oputput:-d1 Numeric 55)
D1=Str(d1)
Disp memo like d1 (oputput:-d1 Character '55')
Function returns:- Character

(5) Left:- This function Returns a specified number of characters from a character expression, starting with the leftmost character.

Syntax:- Left(<exp C>,<exp N>)
Example:- ?Left('computer',4)
Output:- comp
Function returns:- Character

(6) Right:- This function Returns the specified number of rightmost characters from a char string.

Syntax:- Right(<exp C>,<exp N>)
Example:- ?Right('computer',4)
Output:- uter
Function returns:- Character

(**7) Upper :-** This function is use to convert specified character string in UPPERCASE format.

Syntax:- Upper(<exp C>)
Example:- ?Upper('computer')
Output:- COMPUTER
Function returns:- Character

(8) Lower:- This function is use to convert specified character string in lowercase format.

Syntax:- Lower(<exp C>)
Example:- ?Lower('COMPUTER')
Output:- computer
Function returns:- Character

(9) Proper:- This function is use to convert specified character string in Proper case format.

Syntax:- Proper(<exp C>)
Example:- ?Proper('this is a computer')
Output:- This Is A Computer
Function returns:- Character

(10) Ltrim:- This function Returns the specified character expression with leading blanks removed.

Syntax:- Ltrim(<exp C>)
Example:- ?Ltrim('   computer')
Output:- computer
Function returns:- Character

(11) Rtrim:- This function Returns the specified character expression with all trailing blanks removed.

Syntax:- Rtrim(<exp C>)
Example:- ?Rtrim('computer   ')
Output:- computer
Function returns:- Character

(12) Alltrim:- This function Returns the specified character expression with leading and trailing blanks removed.

Syntax:- Alltrim(<exp C>)
Example:- ?Alltrim('   computer   ')
Output:- computer
Function returns:- Character

(13) Isupper():- This function checks whether the first leftmost character of specified expression is in uppercase or not.If it find character in uppercase it return .t. else it return .f. as output.

Syntax:- Is upper(<exp C>)
Example:- ?Isupper('computer')
Output:- .f.
Function returns:- Logical

(14) IsLower():- This function checks whether the first leftmost character of specified expression is in lowercase or not.If it find character in lowercase it return .t. else it return .f. as output.

Syntax:- Islower(<exp C>)
Example:- ?Islower('computer')
Output:- .t.
Function returns:- Logical

(15) IsDigit():- This function checks whether the first leftmost character of specified expression is digit or not. If it find digit it return .t. else it return .f. as output.

Syntax:- Isdigit(<exp C>)
Example:- ?Isdigit('computer')
Output:- .f.
Function returns:- Logical

(16) IsAlpha():- This function checks whether the first leftmost character of specified expression is an alphabetic character or not. If it find character as Alphabet it return .t. else it return .f. as output.

Syntax:- Isalpha(<exp C>)
Example:- ?Isalpha('computer')
Output:- .t.
Function returns:- Logical

(16) Len():- This function Returns the number of characters in a character expression.

Syntax:- Len(<exp C>)
Example:- ?Len('computer')
Output:- 8
Function returns:- Numeric

(17) Substr():- This function Returns a specified number of characters starting from specified character to specified no.of character the given expressions.

Syntax:- Substr(<exp C>,<exp N1>[,<exp N2>])
Example:- ?Substr('computer',1,5)
Output:- compu
Function returns:- Character

(18) Space():- This function Returns a character string composed of a specified number of spaces.

Syntax:- Space(<exp N1>)
Example:- ?Space(10)
Output:- [Ten blank space]
Function returns:- Character

(19) Version():- This function Returns a character string containing the FoxPro version number you are using.

Syntax:- Version()
Example:- ?Version()
Output:- 2.6
Function returns:- Character

(20) Set():- This function Returns the status of a SET command.

Syntax:- Set(<exp C>)
Example:- ?set("talk")
Output:- off
Function returns:- Character

(21) Fsize():- This function Returns the size, in bytes, of a specified field.Before using this function its necessary that the fieldsize that you want to know its database file must opened.

Syntax:- Fsize(<exp C>)
Example:- use student
?Fsize("sname")
Output:- 15
Function returns:- Numeric

(22) Fullpath():- This function Returns the fullpath of specified database file in expression.

Syntax:- Fullpath()
Example:- ?Fullpath("student")
Output:- c:\fpw26\student
Function returns:- Character

(23) Between():-This function checks if the expression1 is lies between two another expression of same datatypes.This function returns true if it find else it return false.

Syntax:- Between(<exp R1>,<exp R2>,<exp R3>)
Example:- ?Between(5,2,10)
Output:- .t.
Function returns:- Logical

(24) Replicate():- This function Returns a character string that contains a specified character expression repeated a specified number of times.

Syntax:- Replicate(<exp C>,<exp N>)
Example:- ?Replicate('*',10)
Output:- * * * * * * * * * *
Function returns:- Character

(25) Inlist():- This function checks whether or not an expression matches one in a series of expressions of the same data type. INLIST( ) returns true if it finds the expression in the set of expressions. If it doesn't find the expression in the set of expressions, INLIST( ) returns false.

Syntax:- Inlist<expr1>, <expr2>[, <expr3> ...])
Example:- ?Inlist(5,10,15,25,35,45,5)
Output:- .t.
Function returns:- Logical

(26) Difference():- This function Returns an integer, 0 through 4, representing the relative phonetic difference between two character expressions.

Syntax:- Difference(<exp C1>,<exp C2>)
Example:- ?Difference("rush","rooos")
Output:- 4
Function returns:- Numeric

(27) Padl():- This function Returns a character string from a character expression padded to a specify length. This function insert padding in leftside.If you not specify padded character computer padded by blank space.

Syntax:- Padl(<expr>, <expN>[, <expC>])
Example:- ?Padl("Jay",20,"_")
Output:- _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ jay
Function returns:- Character

(28) Padr():- This function Returns a character string from a character expression padded to a specify length. This function insert padding in right side. If you not specify padded character computer padded by blank space.

Syntax:- Padr(<expr>, <expN>[, <expC>])
Example:- ?Padr("Jay",20,"_")
Output:- jay_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Function returns:- Character

(29) Padc():- This function Returns a character string from a character expression padded to a specify length in center. This function insert padding in bothsize .If you not specify padded character computer padded by blank space.

Syntax:- Padc(<expr>, <expN>[, <expC>])
Example:- ?Padl("Jay",20,"_")
Output:- _ _ _ _ _ _ _ _ jay_ _ _ _ _ _ _ _ _
Function returns:- Character

Set Commands of FoxPro

Set command are one type of command which is defiantly set by the FoxPro when can change them means set them when we needed.

(1) Set alternate command:-

Set alternate command is use to save your printer output created with ?, ??, DISPLAY or LIST to a text file.

Syntax:- SET ALTERNATE ON | OFF or SET ALTERNATE TO [<file> [ADDITIVE]]
Default clause:- Off

Set alternate to <filename> will create a new text file and stores the data into it.

Example :-
```
Set alternate on
Set alternate to myfile
Use student
list
set alternate off
```

Above example will create a new textfile name as my file and stores all the records of student database file

(2) Set carry command:-

This function Determines whether or not FoxPro carries data forward from the current record to the new record created with APPEND or INSERT. This function is also helpful to carry forward only specific field.

Syntax :-	SET CARRY ON | OFF
	SET CARRY to <Fieldlist>
Default clause:-	Off
Example:-	Set carry on
	Set carry to class,city

(3) Set century command :-

By default the date format of year is in two digit. If we want to display year format as 'yyyy' we can set the century command to ON mode.

Syntax:-	Set century ON|OFF
Example:-	Set century ON
Default clause:-	OFF

(4)Set color command :-

This set command is useful to set Background color and text color in FoxPro. By default white is background color and black is text color is set. We must have to specified picture code of FoxPro which are following.

B---blue	N---Black	G---Green	W---White	I --- Inverce
BG---Cray	GR---Brown	N+---Gray	RB ---magenta	GRT---yellow

Syntax:-	Set color to<text color code>/<Back Color Code>
Example:-	Set color r/G
	clear

Default clause:- Default black textcolor is selected and white is background color.

Above example will set Red as textcolor and Green as background color. We must have to type cleat command to see the output.

(5)Set Deleted command :-

Set deleted command is use to specify whether or not FoxPro process on the records which are marked for deletion when set delete is on. FoxPro does process on marked records when commands like List, Display, Report.

Syntax:-	Set Deleted ON | OFF
Example:-	Set Deleted ON
Default clause:-	OFF

(6)Set Device to command :-

This set command is useful to set the output of @...say command to the specified device that is file, screen or printer. To screen option is use to set output in screen. File <filename> clause is use to save output in another textfile.<filename> specifies the name of file.

Syntax:-	SET DEVICE TO SCREEN | TO PRINTER |
	TO FILE <file>
Example:-	Set device to screen
	Set device to myfile
Default clause:-	By default set default to screen is set.

(7) Set Function command :-

This set command is useful to assign a new value to a function keys. Besides the programming function you can also program combination of function keys other keys such as shift,ctrl,alt.

Syntax:- SET FUNCTION <expN> | <key label> TO [<expr>]

Example:- Set function f9 to list
Set function "alt+f9" to append

<Exp N> specifies the function key number.

<Exp R> specifies the commandname.

Here first example will set list command in F9 function key while second function will set append command in 'Alt+F8'

(8) Set Default to command :-

This set command is useful to set the default drive and Directory for various file read and to write operation.

Syntax:-	SET DEFAULT TO <exp C>
Example:-	Set default to e:\pgdca\roll038
Default clause:-	By default c:\fpw26 path is set in your foxpro

<Exp C> specifies the URL (path of your folder where you want to work)

(9) Set Exact command :-

Set Exact command Specifies the rules used when comparing strings of different lengths. If SET EXACT is ON, expressions must match character for character including blank spaces for the expressions to be equivalent.

Syntax:-	SET EXACT ON | OFF
Example:-	Set Exact on
Default clause:-	Default Clause Is in OFF mode.

(10) Set Exact command :-

Set Exact command Specifies the rules used when comparing strings of different lengths. If SET EXACT is ON, expressions must match character for character including blank spaces for the expressions to be equivalent.

Syntax:-	SET EXACT ON | OFF
Example:-	Set Exact on
Default clause:-	Default Clause Is in OFF mode.

(11) Set Safety command:-

Set Safety command determines whether or not FoxPro displays a warning before overwriting an existing file. If SET SAFETY is ON and you execute a command that overwrites a file, a dialog is first displayed asking if you want to overwrite the file.

Syntax:-	SET SAFETY ON | OFF
Example:-	Set Safety off
Default clause:-	Default Clause Is in ON mode.

(12) Set Date command:-

Set Date command is use to set date format to display date expression. By default the date is display in American format that is 'MM/DD/YY'.

Syntax:- SET DATE TO <Date format>

DATE FORMATS:-

American---	'MM/DD/YY'	British---	'DD/YY/YY'
Japan---	'YY/MM/DD'	German---	'DD.MM.YY'
Ansi ----	'YY.MM.DD'	DMY ----	'DD/MM/YY'
MDY---	'MM/DD/YY'	YMD ----	'YY/MM/DD'

Example:- ?Date()
Set date to British
?Date()

(13)Set Marks to command :-

Set Marks to command specifies a delimiter to use in the display of date expressions.By default ‘ / / ‘is used in date.

Syntax:- SET MARKS TO <exp C>

Example:- ?date() Output:- 01/22/08
Set marks to ‘-‘
?date() Output:- 01-22-08

(14)Set Order to command :-

Set Order command is use to set any index file as master file. When there are more than one index file is created and if you want to set any specific index file as a master file this command is helpful.

Syntax:- SET ORDER TO [<expN1> | <idx index file> |
[TAG] <tag name> [OF <cdx file>]
[IN <expN2> | <expC>]

<expN1> Specifies the number of index file in which order you create.
<expN2> Specifies the work area number in which you want to open your index file.

Example:- Set Order to 2
Default clause:- By Default last created index file remains open.

(15)Set Index to command :-

Set Index command is use to opens one or more index files for use with the current table/.DBF.

Syntax:- SET INDEX TO [<index file list> | ?]
[ORDER <expN> | <idx index file> |
[TAG] <tag name> [OF <cdx file>]

<expN1> Specifies the number of index file in which order you create.

Example:- Set index to n1
Default clause:- By Default last created index file remains open.

(16)Set Printer command :-

Set Printer command is use to Enables or disables output to the printer and routes output to a file or a port. If set device to printer is set and set printer is on the output is redirected to the printer.

Syntax:- SET PRINTER ON | OFF
Example:- Set printer on
Default clause:- Default Clause is in OFF mode.

(17)Set Status command :-

Set Status command is use to display a FoxPro’s own status bar on the screen. By default FoxPro display the windows status bar on the screen But if we want to off the status bar this command is useful.

Syntax:- SET STATUS ON | OFF
Example:- Set status on
Default clause:- Default Clause is in OFF mode

(18)Set Heading command :-

Set Heading command determines whether or not column headings are displayed for fields on commands like List, Display, etc.

Syntax:- SET HEADING ON | OFF
Example:- Set Heading oFF
Default clause:- Default Clause is in ON mode.

(19) Set Fields command :-

Set Feilds command is use to specifies which fields in a Database file can be accessed and whether all or only specified fields can be accessed etc. Only the fields specified in the field list can be accessed when SET FIELDS is ON.

Syntax:- SET FIELDS ON | OFF **or** SET FIELDS TO [[<field1>[,<field2> ..]]| ALL]

Example:-
```
Use Student
Set Fields to sname, stot, sperc
Set fileds on
```
Default clause:- By Default all the fields are display on the screen.

(20) Set Filter command :-

Set Filter command is use to Specifies a condition that records in the current table/.DBF must meet to be accessible.

Syntax:- SET FILTER TO <exp L>

<expL> specifies the condition that records must satisfy to access.

Example:- Set Filter to city=”surat”

Default clause:- Default Clause is OFF

(21) Set Bell command :-

Set Bell command is use to Set set bell to on or off mode. While we enter any wrong entry in any filed computer makes a beep ring called bell. If we want to stop this bell this command is very helpful.

Syntax:- SET BELL ON | OFF

Example:- Set Bell off

Default clause:- Default Clause is ON

(22) Set Clock command :-

Set Clock command determines whether or not FoxPro displays the system clock and specifies the clock location on the screen. When we set clock command to ON mode FoxPro displays the clock in the upper-right corner of the desktop But if we want to change the position we can set by set clock to command. Set clock status command set clock in status bar.

Syntax :- SET CLOCK ON | OFF | STATUS **or** SET CLOCK TO [<row, column>]

Example:- Set clock on

Example:- Set clock to 1,1

Example:- Set clock status

Default clause:- Default Clause is OFF

(23) Set Confirm command:-

Set Confirm command specifies whether or not Enter or Tab must be pressed to exit an input field.

Syntax :- SET CLOCK ON | OFF

Example:- Set Confirm ON

Default clause:- Default Clause is in OFF mode.

(24)Set Scoreboard command :-

Set Scoreboard command specifies Whether FoxPro displays the status of the NumLock, Caps Lock, and Insert keys.

Syntax :- SET SCOREBOARD ON | OFF

Example:- Set Scoreboard off

Default clause:- Default Clause is in ON mode.

(25)Set Message command :-

Set Message command define a message for display under the status bar or specifies the location of messages defined with user-defined menu bars and popup commands.

Syntax :- SET MESSAGE TO [<expC>]

SET MESSAGE TO [<expN> [LEFT | CENTER | RIGHT]]
<expN> specifies the row on the desktop or main FoxPro window on which messages are displayed.
<expC> Specify the message to display in the screen.
Example:- Set message to "jay" right

(26) Set Hours to command :-

Set Hours to command is use to set the system clock to 12 or 24 hours format. By default 12 hours format is set.

Syntax :-	SET Hours to [12 | 24]
Example:-	set hours to 24
Default Clause:-	set hours to 12 (is set)

(27) Set Decimal to command :-

Set Decimal command Specifies the number of decimal places displayed in numeric exp. Set Decimal command is run when we enter set fixed on.

```
Syntax :-     SET Decimals to <exp N>
Example:-     ?5*2.2645
              set Decimals to 2
              set fixed on
              ?5*2.2645
```

(28) Set Fixed command :-

Set Fixed command Specifies whether or not the number of decimal places used in the display of numeric data is fixed.

```
Syntax :-     SET Fixed ON | OFF
Example:-     ?5*2.2645
              set Decimals to 2
              set fixed on
              ?5*2.2645
Default Clause:- Default clause is OFF
```

(29) Set Console command :-

Set Console command is use to Enables or disables output to the screen or a window. Set Console command is always execute in a program file not In a command window.

```
Syntax :-     SET Console ON | OFF
Example:-     Modi comm rpm1
              Set console off
              Store 5 to a,b
              ?"C=",a+b
              set console on
Default Clause:- Default clause is ON
```

(30) Set Relation command :-

Set Relation command is use to create a relation between two different tables in two different work area. Before creating a relation between two tables following points are necessary.

1. Two database files of different name.
2 One is open in workarea1 and second in workarea2
3. Both database file must have one field and on commanfiled indexing is necessary.

```
Syntax :-     SET RELATION TO   [<expr1> INTO <expN1> | <expC1>
                                [, <expr2> INTO <expN2> | <expC2> ...]
Example:-     use student1 in 1
              Use pgstudent1 in 2
              Index on name to nm1
              Set order to 1
              Select 1
```

```
Index on name to nm2
Set order to 1
Set relation to nm2 into 2
Go 5
Disp
Select 2
Disp
Go 10
Disp
Select 1
disp
```

Memory Variable

⇒ Memory variable is also know as a public variable.

Method of Creating Memory Variable:-

We can create memory variable by following three method.

❖ **Store:-**

Syntax:-	Store <expR> to <variablename>
Example:-	Store 0 to m1,m2
	Store {24/01/08} to tdate

❖ **Space:-**

Syntax:-	<variablename>=space(<expN>
Example:-	name=space(15)

❖ **=**

Syntax:-	<variablename>=<expR>
Example:-	name="Rushabh"
	Ch=.t. (For logical value storage)

Explain ?, ?? and ??? command :-

❖ **? Command:-**

This command is known as a print command. Single question marks is use to print any value of memory variable on the screen. Each print command always starts with new line.

❖ **?? Command**

This command is also known as a print command. Two question marks display the expression results on the current line at the current position of the desktop

```
Syntax:-     ? || ??  [PICTURE <expC1>] | [FUNCTION <expC2>]
                      [AT <expN1>]
                      [FONT <expC3>[, <expN2>] [STYLE <expC4> | <expr2>]]
Example:-    ? "name is "+name
Example:-    ?? i
```

❖ **??? Command :-**

Three question marks command is use to send the output direct to the printer.

<u>Clauses of ? || ?? Command:-</u>

PICTURE <expC1>:= If the PICTURE clause is present, the result of <expr1> is displayed according to the format specified by <expC1>. <expC1> can consist of function codes, picture codes or a combination of both.

AT <expN1> :- AT clause is used to specify the column number <expN1> where the output is displayed. The numeric expression <expN1> can be a UDF that returns a numeric value.

FONT <expC3>[, <expN2>]:-

The character expression <expC3> is the name of the font, and the numeric expression <expN2> is the font size. If you include the FONT clause but if you not specified the font size <expN2>, a 10 point font is used. If the font you specify is not available Foxpro automatically select a font with similar font characteristics. The FONT clause is ignored in FoxPro for MS-DOS.

STYLE <expC4> | <expr2>:-

In FoxPro for Windows, you can include the STYLE clause to specify a font style for ? | ?? output. The styles that are available for a font are determined by Windows. If the font style you specify is not available foxpro automatically select a font style with similar characteristics. The font style is specified with <expC4>.

B	Bold	I	Italic	N	Normal	O	Outline
Q	Opaque	S	Shadow	-	Strikeout	U	Underline

You can include more than one character to specify a combination of font styles.

❖ Display Memory command :-

Display memory command is use to show which are the variables are stored in memory. This command also display the data type and data of variable. We can use like option to display only those variables detail which match with the expC.

Example:- Display memory like a

(Above command will display the detail of only those variables list on the screen which are match with starting alphabet 'A'.

❖ Release All command :-

Release all command is use to delete all the variables which are store in RAM. Clear all Command also work same as release command.

❖ Save to command:-

Save to command is use to same all the variables in one file. This file is also known as memory file with the extension of '.mem' Once we store the variable we can restore all the variable in memory using the Restore from command.

Syntax:- Save to <filename.mem>
Example:- save to rpm.mem

❖ Restore command :-

Restore command is use to Restore the variables from the memory file in which you store the variable. You can enter this command only when you already create memory file to save the variable in it. You have to specify the memoryfile name with this command.

Syntax:- Restore from <filename.mem>
Example:- Restore from rpm.mem

Decision making statements:-

(1) If ..endif :-

The **if** statement is a powerful decision making statement and it is used to control the flow of execution of statement. It is a two way decision statement and it used in conjunction with an express.

Syntax:-
```
if (condition)
        Statements
Endif
```
Example:-
```
n=10
If(n=10)
        ?"enter no is ten"
Endif
```

(2) If ..else….endif :-

The **if..else..** statement is a another powerful decision making statement and it is used to control the flow of execution of statement. It is a two way decision statement and it used in conjunction with an expression.

```
Syntax:-        if (condition)
                        Statements
                Endif
Example:-       per=45
                If(per>34)
                        ? "Pass"
                Else
                        ? "Fail"
                endif
```

Do case…..Endcase

The Do case structure allows you to select one set of command from many alternatives.In If..endif you can specify only two alternative while Do case structure allows you to specify as many alternatives as you like.

```
Syntax:-        Do case
                        Case  <expL1>
                                <statements>
                                <break>
                        Case  <expL2>
                                <statements>
                                <break>
                                ……….
                                ……….
                        Case  <expL n>
                                <statements>
                                <break>
                        otherwise
                                <statements>
                Endcase
```

When foxpro enters in Do case..endcase structure it will check the case statements in sequence when it find true condition ,it execute the corresponding command set. If foxpro does not find any case statements as true and structure include the otherwise statements it execute the command given between otherwise and endcase.

Example :- Write a program to check whether the input character is vowel or not.

```
        Ch=space(1)
        Input "Enter only one charater" to ch
        Do case
                Case ch='a' or ch='A'
                        ?"Enter character is vowel"
                Case ch='i' or ch='I'
                        ?"Enter character is vowel"
                Case ch='e' or ch='E'
                        ?"Enter character is vowel"
                Case ch='o' or ch='O'
                        ?"Enter character is vowel"
                Case ch='u' or ch='U'
                        ?"Enter character is vowel"
                otherwise
                        ?"Enter character is not vowel"
        Endcase
```

Looping Constructs

Foxpro provide various types of looping constructs as shown below.

(1) For……endfor
(2) Do while ……enddo
(3) Scan………endscan

(1) For……endfor :

The For…..Endfor executes a set of statements within a loop specified number of times .A memory variable or an array element is used as a counter to specify how many times the statements inside the loop are execute.

```
Syntax:-   FOR <memvar> = <expN1> TO <expN2> [STEP <expN3>]
                   <statements>
                   [EXIT]
                   [LOOP]
           ENDFOR | NEXT
```

<exp N1> specifies the starting position of loop.
<exp N2> specifies the ending position of loop.
<exp N3> specifies the value of stem to loop continue.
<Loop> Loop can be placed anywhere between For…endfor. Loop Returns the control direct to For.
<Exit> Exit can be placed anywhere between For…..endfor. Exit sends the control to out of the For loop and first statement after the Endfor.

```
Example:-   For i=1 to 100                    For i=100 to 1 step -1
                  ?i                                ?i
            endfor                            endfor

            For i=100 to 1 step -5
                  if i%2==0
                        ?i
                  else
                        loop
                  endif
            endfor
```

The First example will print 1 to 100 on the screen. The second example will print 100 to 1 on the screen. while third example will print 100,90,80,70,60,50,40,30,20,10 on the screen.

(2) Do while……enddo:-

Do while command is very powerful and flexible set of command and it is used to execute the command repeatedly. Do while command executes a block of statements within a conditional loop.

```
Syntex:-    DO WHILE <expL>
                  <statements>
                  [LOOP]
                  [EXIT]
            ENDDO
```

A set of statements is placed between Do while and enddo are execute as long as the logical expression <expL> remains true. Do while statement must have a corresponding enddo statement.

<exp L>	specifies the logical expression. The statement placed between do while and enddo are execute as long as this <exp L> evaluates to true.
<Loop>	Loop can be placed anywhere between Do while…..enddo. Loop Returns the control directly back to Do while.
<Exit>	Exit can be placed anywhere between Do while…..enddo. Exit sends the control to out of the Do while loop and first statement after the enddo.

Example:-

```
i=1                         x=10
do while i<100              do while x>0
     ?i                          if x%2==0
     i=i+1                            ?x
endfor                           else
                                      loop
                                 endif
                                 x=x-1
                            enddo
```

The First ex. will print 1 to 100 on the screen. The second example will print 10,8,6,4,2 on the screen.

(3) Scan……endscan:-

Scan….endscan is another powerful structure. A condition is specified with scan and foxpro execute the statements included within a Scan and Endscan on all the records of the current open database file that meet the condition.

Syntax :-

```
SCAN  [<scope>]
      [FOR <expL1>]
      [WHILE <expL2>]
      [<statements>]
      [LOOP]
      [EXIT]
ENDSCAN
```

<Statements> specifies the block of statements execute between Scan and Endscan.

[For<condition>]:- Including the FOR clause lets you filter out undesired records. If the FOR clause is included, the commands are executed for all records within the scope for which <expL1> is true.

<scope> :- The scope clauses are: ALL, NEXT <expN>, RECORD <expN>, and REST. You can specify a scope of records that are scanned. Only the records that fall within the range of records specified by the scope are scanned.The default scope for SCAN is ALL records.

WHILE <expL2> : If the WHILE clause is included, the commands are executed as long as <expL2> remains true.

<Loop> : Loop can be placed anywhere between Scan …..endscan. Loop Returns the control directly back to Scan.

<Exit> : Exit can be placed anywhere between Scan…..endscan. Exit sends the control to out of the Scan and first statement after the endscan.

Example:-

```
use address
Scan city=”surat”
     Display
Endscan
```

Above example will display the record of person who lives in surat of address database file. we have no need to skipping the cursor to the next record. can automatically check all the records because default scope of scan is all record.

Arrays

- An array is a set of related data item.
- An array is a group of variable having same name but with different index value.

There are three types of array

1. **One dimensional Array**
2. **Two dimensional Array**
3. **Multi dimensional Array**

Note:- In Foxpro you have to learn only one and two dimensional arrays. And the simple concept of multidimensional array.

- ⇒ You can create any Array variable using 'Declare' keyword or using 'Dimension' keyword.
- ⇒ In foxpro the index value of array is start with 1 not with 0.
- ⇒ In foxpro you not have to declare the Data type of array variable.
- ⇒ Datatype of array variable is depends on its value.
- ⇒ We can use '[ ]' bracket of '()'brackets to declare the size of array variable.
- ⇒ The default datatype of array variable is logical
- ⇒ We can change the size of array at any time like increment after declaration.

Another Database Related functions

(1) BOF() :- This function checks whether the cursor is at the beginning of database file or not. If it found the cursor at the beginning of file it return true (.T.) as answer else it return False(.F.) as answer.

Example:-	use student
	?BOF()
Output:-	**.f.**
Return:-	Logical

(2) EOF():- This function checks whether the cursor is at the end of database file or not. If it found the cursor at the end of file it return true (.T.) as answer else it return False(.F.) as answer.

Example:-	use student
	?EOF()
Output:-	**.t.**
Return:-	Logical

(3) Found():- This function returns true (.T.) if continue, find, locate or seek is successful otherwise it return false.

Syntax:-	Found([<expN> | <expC>])
Return:-	Logical

(4) Afield() :- This function is use to store table structure information into an array variable. AFIELDS( ) places information about the structure of the current table into an array and returns the number of fields in the table. AFIELDS( ) stores each field name, type, length, and the number of decimal places in numeric fields.

Syntax:-	AFIELDS(<array>)
Returns:-	Numeric

(5) Fcount():- This function returns the number of fields in the current or specified table/.DBF.

Syntax:-	FCOUNT([<expN> | <expC>])
Returns :-	Numeric

<expN> | <expC>:-

You can return the number of fields in a table/.DBF open in another work area by specifying the work area number <expN> or the table/.DBF or work area alias <expC>. 0 is returned if a table/.DBF isn't open in the work area you specify. If you don't specify a work area or alias, the number of fields is returned for the table/.DBF open in the selected work area.

(6) Recno() :-

This function returns the current record number from the current or specified table/.DBF.If database file has no record it return 1 and Eof() function return true.<expN> or <expC> is use to specify the workarea name or workarea number.

Syntax:- RECNO([<expN> | <expC>])
Returns:- Numeric

(7) Reccount() :-

This function returns the number of record in the table. This function ignores the state of Set deleted command and Set filter command.

Syntax:- RECCOUNT()
Returns:- Numeric

Functions Related to Advance Foxpro Programming:-

(1) Pad() :-

Pad function returns the name of menupad that was most recently choosen from a menubar.

Syntax:- Pad()
Function Returns:- Character
Example :- ?pad()

(2)Parameter() :-

Parameter function returns the number of parameter that were passed most recently called program, procedure or in a user define function.

Syntax:- Parameter()
Function Returns:- Numeric
Example :- ?pad()

(3) Prompt() :-

Prompt function the prompt text of a menu pad choosen from a menu bar or an option choosen from a popup.

Syntax :- Prompt()
Function Returns :- Character
Example :- ?pad()

(4)Popup() :- Popup function returns the name of active popup.

Syntax :- Popup()
Function Returns :- Character
Example :- ?popup()

Command Related to Programming in FoxPro

(1) Input command :-

Input command is use to input any data from user and to store in any particular memory variable or in array variable. We can also display a message for entering data using <exp C> clause.

Syntax:- Input <exp C> to <memory variable>
Example:- Input “value of A : “ to a

(2) Accept command :-

Accept command is input any data from user and to store in y particular variable memory variable or in array variable.The difference between input and accept command is that input command is use to input any numeric data only while accept is use to input string type of data. We can enter string type data using input command by entering string in double quotation marks.

Syntax:- Accept <exp C> to <memory variable>
Example:- Accept "Enter name:- " to name

(3)@ ….Say…. command:-

This command is use to display any output at specified column.It can also be used to format output for a printer.

```
Syntax :-    @ <row, column> SAY <expr>
                [FUNCTION <expC1>]
                [PICTURE <expC2>]
                [SIZE <expN1>, <expN2>]
                [FONT <expC3>[, <expN3]]
                [STYLE <expC4>]
                [COLOR SCHEME<expN4>|COLOR<colorpair list>]
```

We can use combine @..say .. command and @ ..Get command into a single command. If both the SAY and GET clauses are included, specify a single set of coordinates <row, column> where @ ... SAY output begins. A space is automatically inserted between the @ ... SAY output and the @ ... GET text editing region.

<row, column> clause:-

Rows and column are numeric expression with values 0 and greater, that determine where the output of @ …Say .. is appear. The first row is 0 number. Rows are numbered from top to bottom. The first column is number is 0 on the desktop. Columns are numbered from left to right.(In foxpro there are about 1 to 40 rows and 1 to 100 columns are available in general pc.)

@ <row, column> SAY <expr>:-

<expR> is evaluated and display or printing at <row,column> position.<expR> can be string, name of memory variable.

FUNCTION <expC1> | PICTURE <expC2> clauses:-

You can include the FUNCTION clause, the PICTURE clause or both to control how <expr> is displayed or printed.

FUNCTION Code	Purpose
B-----	Left-justifies numeric data within the display region.
C-----	CR is displayed after a positive number to indicate a credit. Can be used with numeric data only.
D-----	Uses the current SET DATE format.
E-----	Edits date type data as a BRITISH date.
T-----	Trim leading and trailing blanks from <expr>.
X-----	DB is displayed after negative numbers to indicate a debit. Use only with numeric data.
Z------	<expr> is displayed as all blanks if its numeric value is 0. Use only with numeric data.(Encloses negative numbers in parentheses. Use only with numeric data.
!------	Converts alphabetic characters to upper-case. Use with character data only.
^ ------	Displays numeric data using scientific notation. Use with numeric data only.
$------	Displays data in currency format. The currency symbol appears before or after the field value depending on the current setting of SET CURRENCY. Use with numeric data only.

A PICTURE expression can include any characters, but only the characters listed below actively participate in display and printing.

PICTURE Code	Purpose
X-----	Allows any character.
Y-----	Allows logical Y, y, N and n only. Converts y and n to Y and N, respectively.
!------	Converts lower-case letters to upper-case letters.
$-----	Displays the current currency symbol. By default, the symbol is placed immediately before or after the field.
*------	Asterisks are displayed in front of the numeric value. Use with a dollar sign $ for check protection.
.	A decimal point specifies the decimal point position.
,	A comma is used to separate digits to the left of the decimal point.

We can use both the combination of function clause and picture clause in one command.

SIZE <expN1>, <expN2> clause:-

The SIZE clause is use to control the length and height of a @ ... SAY display or print region. FoxPro creates a region one row high by default. Include the optional SIZE clause to create a region that is more than one row high. The height of the region in rows is specified by <expN1>, and the width in columns by <expN2>. The @ ... SAY output word wraps at the width specified by <expN1> for <expN2> rows.

In FoxPro for Windows, the @ ... SAY font determines the size of the editing region. The @ ... SAY font is specified with the FONT clause. If the FONT clause is omitted, the @ ... SAY output uses the font of its parent window (the main FoxPro window or a user-defined window)

Example:- @ 2,15 say “jay hind” func “!”

FONT <expC3>[, <expN3>] clause:-

You can use font clause to change the font and its style to be display in output. The character expression <expC3> is the name of the font, and the numeric expression <expN3> is the font size.1Default Font size is 10.

Example:- @ 2,15 say “Jay Hind” font “arial”,18

STYLE <expC4>

In FoxPro for Windows, include the STYLE clause to specify a font style for @ ... SAY output. The styles that are available for a font are determined by Windows. If the font style you specify is not available, Windows substitutes a font style with similar characteristics. The font style is specified with <expC4>. If the STYLE clause is omitted, the standard font style is used.

Character	Font Style						
B	Bold	I	Italic	N	Normal	O	Outline
Q	Opaque	S	Shadow	-	Strikeout	T	Transparent
U	Underline						

You can include more than one character to specify a combination of font styles. For example, the following command specifies Bold Italic:

Example:- @ 2,15 say “jaydev” style “BI”

COLOR SCHEME <expN4> | COLOR <color pair list> clause:-

If you do not include a COLOR clause, the color of @ ... SAY output is determined by the color scheme for the desktop or the main FoxPro window. Only the first color pair in a color scheme or color pair list affects the color of @ ... SAY output. The color of @ ... SAY output can be specified by including the number of an existing color scheme in the COLOR SCHEME clause or a set of color pairs in the COLOR clause.

A color scheme is a set of 10 predefined color pairs. The color pairs in a color scheme can be changed with SET COLOR OF SCHEME.

A color pair is a set of two letters separated by a forward slash. The first color letter specifies the foreground color and the second letter specifies the background color.

Example:- @ 15,15 SAY ' hello! ' COLOR G/B

(4) @ ….get… command:-

This command is use to input any value of the user at t at specified column.@ …get command is work when read statement is written after the get command.

```
Syntax :-    @ <row, column> GET <memvar> | <field>
                 [FUNCTION <expC1>]
                 [PICTURE <expC2>]
                 [FONT <expC3>[, <expN1>]]
                 [STYLE <expC4>]
                 [DEFAULT <expr1>]
                 [ENABLE | DISABLE]
                 [RANGE [<expr2>] [, <expr3>]]
                 [SIZE <expN2>, <expN3>]
                 [VALID <expL1> | <expN4> [ERROR <expC6>]]
                 [WHEN <expL2>]
                 [COLOR SCHEME<expN5>|COLOR<colorpair list>]
```

Note:- Here many clause of @ ..get ..command works same as work in @ … say .. command. so I am nor repeating those clause please refers from @ ..say…command.

DEFAULT <expr1>

If you specify a memory variable for the @ ... GET editing region that doesn't exist, it is automatically created and initialized if you include DEFAULT .However,an array element isn't created if you specify an array element in a DEFAULT clause. The DEFAULT clause is ignored if the memory variable already exists or you specify a field.

Note:- If the DEFAULT clause isn't included and the memory variable <memvar> doesn't exist, the error message "Variable not found" is displayed.

The DEFAULT expression <expr1> determines the type of memory variable created and its initial value.

ENABLE | DISABLE clause:-

Including DISABLE prevents access to an @ ... GET editing region. The editing region is displayed in the disabled colors and cannot be selected. By default, @ ... GET editing regions are enabled. You can include ENABLE as a reminder in a program that a GET editing region can be accessed.

RANGE [<expr2>] [, <expr3>]

Use the RANGE clause with character, date and numeric data to specify a range of acceptable values. If the value you enter in the @ ... GET editing region isn't within the specified range, a message showing the correct range is displayed. The lower boundary of the range is specified with <expr2>, the upper boundary with <expr3>. <expr2> and <expr3> must be character, numeric or date expressions that correspond to the data in the memory variable, array element or field. Either <expr2> or <expr3> can be omitted, but not both. If one boundary is omitted, the data you enter is checked against the specified boundary only.

VALID <expL1> | <expN4>

Use VALID to validate input. When you attempt to exit the GET editing region, the VALID expression is evaluated. A VALID clause greatly simplifies data validation when used with a user-defined function (UDF). If a UDF is called within a VALID clause in @ ... GET, the UDF should return a logical or numeric value.

<expL1> If<expL1>evaluates to a logical true the input is considered correct and the editing region is exited.If <expL1> evaluates to false the value you entered is considered incorrect and a message is displayed directing you to reenter the data after pressing the Spacebar.

If a UDF validation routine performs a replacement on a field and the validation routine then returns false, the field replacement occurs but the previous field value is restored when you return from the UDF to the GET editing region.

<expN4> A VALID clause that includes a numeric expression is used to specify which object is activated after you exit the GET editing region.

WHEN <expL2> clause:-

WHEN allows or prohibits access to a GET editing region based on the value of <expL2>, which must be true (.T.) before the GET editing region can be accessed. If WHEN is specified and <expL2> is false (.F.), the GET editing region cannot be accessed and the next object is activated.

(5) @...... Box .. command:-

This command is use to Draws a box using specified Coordinates.

```
Syntax:-        @ <row1, column1>, <row2, column2> BOX
                [<expC>]
```

<row1, column1>, <row2, column2> Clause:-

<row1, column1> are the coordinates of the upper-left corner of the box, and <row2, column2> are the coordinates of the lower right corner of the box. If row1 and row2 are the same, a horizontal line is drawn. If column1 and column2 are the same, a vertical line is drawn.

<expC>

The characters used to draw the box can be specified by including <expC>, the character expression, which can include nine characters one for each corner, one for each side and one to fill the box. The characters are displayed starting with the upper-left corner and continuing clockwise. If a ninth character is specified, it is used to fill the interior of the box. If just one character is specified, it is used to draw the entire border. If <expC> is omitted, the box is drawn using a single-line.

(6) @...... to….. command:-

This command is use to Draws a box, circle, or ellipse using specified coordinates. Use this command to draw a box. If you omit the optional clauses, the box is drawn with a single line for the border and the current colors.

```
Syntax:-        @ <row1, column1> TO <row2, column2>
                        [DOUBLE | PANEL | <border string>]
                        [PATTERN <expN1>]
                        [PEN <expN2>[, <expN3>]]
                        [STYLE <expC>]
                        [COLOR SCHEME<expN4>|COLOR<colorpair list>]
```

DOUBLE | PANEL | <border string> Clause:-

If DOUBLE is included, the box is drawn with a double-line border. If PANEL is included, the box is drawn with a solid border.

<border string> is a set of characters that specify parts of the box in this order: top, bottom, left side, right side, upper-left corner, upper-right corner, lower-left corner, lower-right corner.

Diplay	List
1. Default scope of Display command is current record.	1. Default scope of List command is all record.
2. Display command displays the record pagewise.	2. List command does not displays the record page wise.
3. Display command displays the field headings when there are more than one page record.	3. List command does not displays the field headings when there are more than one page record.
4. Display command displays the records which are marked for deletion when set deleted is on.	4. List command displays the records which are marked for deletion when set deleted is on.

Edit	Browse
1. Edit command will display the output in the format of Row wise that is like a append screen.	1. Browse command will display the output in the format of Row and column wise that is like a table.
2. The edit command does not have width option.	2. The browse command have width option.
3. Edit command is little bit difficult because for each and every modification you need to reach to a particular record. So it is time consuming.	3. As you can show all the record at a time, it is easy process for modification.
4.Edit command does not support toggle delete or to insert a new record.	4. Browse command allows you to delete record as well as to append record.

Sorting	Indexing
1. Sorting a database file create a new database file.	1. Indexing a database file creates an index file.
2. Sorting help us to search the desired information quickly.	2.The indexfile helps us to search information where a particular record is available.
3. Sort command is automatically attached with '.dbf' extension.	3. Index command is depand on type to attached with '.ecx' or '.idx' extension,
4. The sort file is bigger in size than the index file.	4.The index file is smaller in size than the sorted file.
5. Sorting is a slow process.	5. Indexing is a faster process.